Tales from Building a SQL Server Data Warehouse in Azure

Experiences & lessons learned from a migration to Azure

August 21, 2017

Lead Analytics Architect, SentryOne
Microsoft Data Platform MVP

Melissa Coates

Blog: www.sqlchick.com
Twitter: @sqlchick
Goals for This Session

1. Share decisions and lessons learned from a recent Azure implementation
2. Introduce key concepts for a deployment to Azure
3. Discuss items involved with building a DW environment in Azure

Azure services & features change frequently.
The information in the presentation is accurate as of mid-August 2017.
Agenda

- Key Azure Concepts
- Deciding on Azure VM vs. Azure SQLDB vs. Azure SQLDW
- Additional Planning Considerations
- Building the Azure Environment
  - Structuring Dev, Test, & Prod Environments
  - Naming Conventions & Tags
  - Configuration Decisions
  - Automation & Scheduling
  - Monitoring the Solution
- Key Takeaways & Open Q&A

Out of scope:
- Linux deployments
- Azure Stack & private cloud deployments
- On-premises physical & virtualized deployments (including Fast Track DW & APS/PDW)
- Security in depth (though we touch on a few points)
- Networking & connectivity
- Troubleshooting, performance tuning & growth
- Details on how to install and configure SQL Server
Key Azure Concepts
The Azure Lingo

Azure Directory

Subscription

Resource Group

Resource

Resource

Resource

Resource Group

Resource Group

Subscription

Subscription

Subscription

Resource Group

Azure Resource Manager (ARM)

Azure Active Directory

https://azure.microsoft.com/en-us/overview/what-is-cloud-computing/

Active Directory

Office 365

Power BI
Resource Groups

We have learned:
✓ Keep resource groups more narrow than broad
✓ Select the region (location) carefully

Planning for resource groups is critical

Focus on:
- Logical organization
- Permissions
- Policies

Scope of ARM automation scripts (exception: Resource Explorer)
Examples – Resources & Resource Groups

IaaS

Azure Resource Group: InternalReportingRGProd

Virtual Machine: SQL Server Engine, SSIS, MDS

Source Data

API

Resource Group: InternalReportingRGTest

Resource Group: InternalReportingRGDev

Storage Account

SQL Server Data, Log, TempDB

SQL Server Backups

SQL Server Diagnostics, Counters, Event Logs

Virtual Machine: SQL Server Engine, SSAS MD

Key Vault

Virtual Machine: SQL Server Engine, SSAS MD

Recovery Vault

Storage Account

SQL Server Data, Log, TempDB

SQL Server Backups

SQL Server Diagnostics, Counters, Event Logs

PaaS

Azure Resource Group: SQLSkills Waits Library

Azure SQL Database

Web App

SQLSkills Waits Library

Cache

Backup & Logging

Storage Account

App Service Plan

Alerts

Alerts
IaaS vs. PaaS vs. SaaS

**Public Cloud**
- Infrastructure as a Service (IaaS)
  - SQL Server in a Virtual Machine
- Platform as a Service (PaaS)
  - Azure SQLDB & Azure SQLDW
- Software as a Service (SaaS)
  - Power BI, Office 365

**On-Premises**
- Physical Server
- Virtual Server

**Highly Scalable**
- Azure SQLDB
- Azure SQLDW
- Azure Stack (Private Cloud)

**Limits to Scalability**
- SQL Server in a Virtual Machine
- Azure SQLDB & Azure SQLDW
- Power BI, Office 365

**Shared Infrastructure (Lower Cost)**
- Less Control (Lower Administration Effort)

**Dedicated Infrastructure (Higher Cost of Ownership)**
- More Control (Higher Administration Effort)
Some resources scale, or even pause, compute separately from storage.
Deciding on Azure VM vs. Azure SQLDB vs. Azure SQLDW
Comparing the SQL Offerings in Azure

**SQL Server in a Virtual Machine (IaaS)**
- Run full workload within an Azure virtual machine, including SQL Server, SSIS, SSAS, SSRS, etc.

**Azure SQL Database (PaaS)**
- A relational database-as-a-service (DBaaS)
  - Non-Managed
  - Managed Instance (Newer - closer feature parity to SQL Server (instance level features))

**Azure SQL Data Warehouse (PaaS)**
- An data warehouse-as-a-service (DWaaS) optimized for performance and large-scale distributed workloads
- MPP architecture (massively parallel processing)
## Comparing the SQL Offerings in Azure (2/2)

<table>
<thead>
<tr>
<th>SQL Server in a Virtual Machine (IaaS)</th>
<th>Azure SQL Database (PaaS)</th>
<th>Azure SQL Data Warehouse (PaaS)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Best for:</strong></td>
<td><strong>Best for:</strong></td>
<td><strong>Best for:</strong></td>
</tr>
<tr>
<td>✓ Migrating existing solutions</td>
<td>✓ New database solutions</td>
<td>✓ DW with larger data volumes</td>
</tr>
<tr>
<td>✓ Running any software and/or all SQL Server features</td>
<td>✓ OLTP with scaling &amp; pooling needs</td>
<td>(bare min. of 1-4TB)</td>
</tr>
<tr>
<td>✓ Administering all aspects</td>
<td>✓ Reduced administration of DB, OS, HA, and DR</td>
<td>✓ Ability to scale compute up/down, or pause (elasticity)</td>
</tr>
<tr>
<td>✓ Bring your own license (Software Assurance)</td>
<td>✓ SLA: for the database</td>
<td>✓ Data Lake Store integration (relational + nonrelational data)</td>
</tr>
<tr>
<td>✓ Isolated dev/test environments</td>
<td></td>
<td>✓ Reduced administration</td>
</tr>
<tr>
<td>✓ SLA: for the VM</td>
<td></td>
<td>✓ SLA: for the database</td>
</tr>
</tbody>
</table>

Many features go first to Azure SQLDB (“cloud first”). However, there are some key features not available in SQLDB (PaaS):
- PolyBase
- R Services
- Change data capture
- CLR
- DB snapshots
- Some T-SQL syntax
- Profiler
- Non-primary filegroups

Also, some features rely on Premium edition:
- Columnstore indexes

Full list: https://docs.microsoft.com/en-us/azure/sql-database/sql-database-features
Key Differences with Azure SQL Data Warehouse (1/2)

- Control Node
- Compute Node
- Compute Node
- Compute Node
- Data Storage

- MPP Architecture
- Shared-Nothing Architecture
- Decoupled Storage & Compute
- Scale Up, Down, Pause
- PolyBase
Key Differences with Azure SQL Data Warehouse (2/2)

- Not All Features Supported
- Different Data Loading Patterns
- Distribution Keys are Critical
- Denormalized Data Model is Best

Take time to educate yourself on the key differences with the MPP architecture—it will affect the design & the data load processes.
Our Decisions on What to Use

SQL Server in a Virtual Machine (IaaS)
- We are using a VM for:
  - ✓ SQL Server DW
  - ✓ Integration Services
  - ✓ Analysis Services (MD)
  - ✓ Master Data Services
  - ✓ R Services

Azure SQL Database (PaaS)
- We are using SQLDB for:
  - ✓ A specific use case: public reporting solution via SQLSkills Waits Library
  - ✓ This SQLDB is loaded from the DW (in SQL Server)

Azure SQL Data Warehouse (PaaS)
- We put SQLDW on roadmap:
  - ✓ Future data growth
  - ✓ Future PolyBase integration with multi-structured data in Azure Data Lake Store

We need VMs for SSIS and SSAS anyway, so we couldn’t justify migrating the relational DW to a PaaS solution at this time

Requires some refactoring of the data load processes so we are planning the move to SQLDW strategically
Our Current State

IaaS

Source Data

API

Storage Account

SQL Server Data, Log, TempDB

SQL Server Backups

SQL Server Data, Log, TempDB

SQL Server Backups

Azure Resource Group: InternalReportingRGTest

Azure Resource Group: InternalReportingRGProd

Virtual Machine: SQL Server Engine, SSIS, MDS

Key Vault

Virtual Machine: SQL Server Engine, SSAS MD

Recovery Vault

Power BI

PaaS

Azure Resource Group: PaaS

SQL Skills Waits Library

Cache

SQL Skills Waits Library

Azure SQL Database

Web App

Storage Account

App Service Plan

Azure Resource Group: Monitoring and Alerts

Alerts

Azure Resource Group: MonitoringAndAlertsRGProd

Alerts
Additional Planning Considerations
Our Goals & Requirements for the Move to Azure

- Expand infrastructure to support future growth
- Ensure ability for Analytics Team to manage environment independently
- Support existing solutions with little to no redesign or refactoring
- Secure connectivity via VPN
- Minimize cost where practical
- Acceptable performance of hourly ETL jobs

Think about trade-offs you’re willing to make for cost, performance, security, regulatory compliance, DR/backups/redundancy, and simplicity
Initial Planning Before Provisioning Any Resources  (1/3)

**First Steps**
- Licensing and Editions
- Full Cloud Implementation vs. Hybrid (Partially On-Premises)
- IaaS vs. PaaS decisions; feature comparison

**Capacity Planning & Cost Estimates**
- Big 3: Storage, memory, CPU
- Networking
- Scalability needs

**High Availability & Disaster Recovery**
- Down time sensitivity (RPO, RTO)
- SLAs from Azure

**Compliance & Security**
**Initial Planning Before Provisioning Any Resources  (2/3)**

- **Domain service accounts & credentials**
  - How administrative & owner permissions will work if decentralized
  - Read and/or write permissions for source or related systems
  - One domain service account per service, per environment
  - Sync to Azure Active Directory for domain users & groups
  - Service principals for certain resources in Azure are very important

- **Azure region: primary location & data redundancy**
  - Geographic location of data
  - Proximity to business users
  - Co-location of related resources
  - Minimizing latency
  - Minimizing data egress charges (very inexpensive though)
  - Not all resources/services are available in every region
We went with a backup/restore approach for the SQL Server files for the DW.

The exception to this was SSISDB custom auditing & logging objects. This DDL was deployed from SSDT after the SSISDB catalog was configured.

Info on migration techniques:

Azure SQL Data Warehouse migration utility:
Structuring Dev, Test & Prod Environments
Separation of Dev, Test, Prod Environments

Most commonly environments are segregated by:
1. Resource Groups,
2. Subscription, or
3. Directory, or
4. A combination of 1 and 3, or 2 and 3

For large or multi-tenant implementations: be aware of Azure limits before deciding.
Option: Separate By Directory

Pros:
✓ Clear boundary
✓ Offers the most scalability

Cons:
✓ More infrastructure to manage
✓ A lot of objects intermixed in a subscription - need clear resource group names and/or tags to tell what belongs to which team
Option: Separate By Subscription

Pros:
✓ Clear boundary

Cons:
✓ Cannot always provision new resources if you’re just an owner at the resource group level
✓ Separate virtual networks & VPN set up
✓ A lot of objects intermixed in a subscription - need clear resource group names and/or tags to tell what belongs to which team
Option: Separate by Resource Group

Pros:
- ✓ Isolation of subscriptions by the team who owns/manages
- ✓ Set up of virtual network & VPN just once
- ✓ Co-admin privileges easier to delegate at the subscription level for each team
- ✓ Billing segregation
- ✓ Less likely to hit Azure limits

Cons:
- ✓ More risk because Prod is mixed in with Dev & Test (mitigate w/ diff IDs to access Prod resources)

We are currently using this option.
Naming Conventions & Tags
Naming Conventions

Purpose → Type of Service → Environment

✓ Type of service in the name helps with logging/metrics in monitoring scenarios

✓ Environment as the suffix makes any concatenations easy within scripts

✓ Prod is enumerated because we work within one subscription

✓ No dashes since all services don’t allow them

✓ Camel case if the service allows it; otherwise lower case

Standard Tags

Tags are very helpful for:

✓ Billing or cost center categories
✓ Environment names
✓ Project or system
✓ Purpose or application
✓ Team, department, business unit
✓ Who owns or supports a resource
✓ Release or version #s (ex: testing infrastructure)
✓ Archival date (ex: if needed only temporarily)
✓ Patching or maintenance window or SLA
✓ Which customer it applies to (ex: if an ISV)
✓ etc...


Try to assign tags right away so you don’t have partial billing without tags assigned
Configuration Decisions:
Azure SQL Database
Use of Azure SQLDB

We’re using Azure SQL Database for a new solution just created recently.

Our solution contains aggregate, anonymous data.

There is no Virtual Network/VPN functionality for SQLDB or SQLDW yet (coming in “Managed Instances” of Azure SQLDB).
Firewall for Azure SQLDB

A firewall can be set at the server level, or at the database level.

Server level firewall rules can be set in the portal:

Database level firewall rules must be set with T-SQL. Although that makes them less visible, it is a better practice.

Try to assign a name to each entry. The generic default name isn’t helpful.
Configuration Decisions:
SQL Server in Azure Virtual Machine
Starting Point: VM Image

There's not currently a VM image which is Fast Track certified. This sets storage disk stripe size to 256KB and trace flags 610 & 1117.

Most of the guidance in this Performance Best Practices article is implemented in the pre-built image.  
Changes Made to Pre-Built VM Image

We started with a pre-built image, and made customizations after it was domain-joined, such as:

- Domain service accounts for each service
- Disable unused services
- Disk structure + permissions
  - G: Data
  - L: Logs
  - T: TempDB
- Enable disk encryption
- Policies (volume maintenance, etc)

SQL Server changes for each new VM:
- Enable the remote DAC
- Default database file locations
- Fill factor settings at database level
- Optimize for ad hoc workloads
- Max degree of parallelism
- Cost threshold for parallelism
- Startup parameters & trace flags
- Expand # of TempDB files & relocate
- Time of SSIS maintenance job
- Default for backup checksum
- Limited account to act as DBOwner etc...
The VHDs (virtual hard disks) behind a VM are in Azure Storage.
Managed/Unmanaged
We chose to use unmanaged storage as shown above (managed was released to preview during our implementation). Managed VM storage doesn’t display in the portal.

Premium/Standard
We are using standard storage for VMs in Dev and Test, and Premium in Production.
Default disks provisioned by the pre-built VM image.

We omitted the F: drive (after moving TempDB to its proper drive).

The default data disk size is 1TB

To release the lease on the storage blob:
1. Delete the volume, and
2. Detach disk from the VM

Don’t forget to delete the file in Azure Storage as well.
VM Disks

We are using **Storage Spaces (aka Virtual Disks)**: multiple Azure data disks in a storage pool to collectively share throughput and IOPS, limits.

This aggregation of disks is helpful due to lower Standard limits on IOPS & throughput.

(Another alternative to increase disk performance is to spread separate database files on different disks.)

\[
\text{IOPS} = \text{# of requests per second (OLTP)}
\]

\[
\text{Throughput (Bandwidth)} = \text{IOPS x I/O Size per specified interval (DW)}
\]

VM Disks

Step 1: Check max # of disks which can be attached to the VM

Step 2: Add disks to VM (Azure portal or PowerShell)

Step 3: Add storage pool with disk striping
  Stripe size (virtual disk interleave) for DW workload: 256K
  Volume allocation unit size: 64K

Step 4: Create folders + reassign SQL Server pathing to your preference
Step 2: Add disks to VM (Azure portal or PowerShell)

Note the single disk performance before it’s collectively shared.
VM Disks

Step 3: Add storage pool + virtual disk + volume in the VM

Step 4: Create folders + reassign SQL Server pathing

G:\MSSQL\Data
G:\MSSQL\Backup
G:\OLAP\Data
G:\OLAP\Backup
L:\MSSQL\ErrorLog
L:\MSSQL\Log
L:\MSSQL\Dump
L:\OLAP\Log
L:\OLAP\Dump
T:\MSSQL\Data
T:\OLAP\Temp
End Result: 8 data disks attached to the Azure VM

Disks attached to the VM

Blob storage

We’re going to handle encryption next.
End Result: 3 storage pools

Stripe size should be 256KB for a data warehousing workload
VM Disks

End Result:
3 volumes
G:\ - Data
L:\ - Log
T:\ - TempDB
Encryption – Data at Rest

Storage Service Encryption (SSE) ← associated with the storage account

Disk Encryption (BitLocker) ← inside the VM

More info:


Other options to consider for encrypting the database:
TDE (Transparent Data Encryption)
AE (Always Encrypted)
Disk Encryption

Step 1: Create a Service Principal

Step 2: Provision Azure Key Vault

Step 3: Enable the Service Principal to communicate with Key Vault

Step 4: Enable Disk Encryption

PowerShell scripts here:


End Result: All drives are encrypted.

A new Bek volume is created which is where the encryption key is read from:

Disaster Recovery

We (the Analytics Team) don’t want to incur the cost or complexity of supporting an availability set. Therefore, we are doing the following for DR purposes:

1. Documented steps for re-provisioning & redeploying

2. Use of Premium storage in Production – this gets us a 99.9% uptime SLA for a single instance Azure VM

3. For Prod: we are considering use of Azure Site Recovery (still in preview)
   [https://docs.microsoft.com/en-us/azure/site-recovery/site-recovery/](https://docs.microsoft.com/en-us/azure/site-recovery/site-recovery/)

   **Important to create your Recovery Services vault in a different region (the RG and the vault itself) than the source objects**
Automation & Scheduling
# Daily & Weekly Schedule

## Production:

<table>
<thead>
<tr>
<th>Time</th>
<th>Task Description</th>
<th>Start/End Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>12am</td>
<td>Engine+SSIS VM shut down</td>
<td>Auto start</td>
</tr>
<tr>
<td>1am</td>
<td>ADF job</td>
<td>Daily ETL</td>
</tr>
<tr>
<td>2am</td>
<td>Hourly ETL &amp; SSAS processing</td>
<td></td>
</tr>
<tr>
<td>3am</td>
<td>EST standard business hours</td>
<td></td>
</tr>
<tr>
<td>4am</td>
<td>UK standard business hours</td>
<td></td>
</tr>
<tr>
<td>5am</td>
<td>Auto patching</td>
<td>(Fri)</td>
</tr>
<tr>
<td>6am</td>
<td>SSIS maint</td>
<td>(Daily)</td>
</tr>
<tr>
<td>7am</td>
<td>SQL backup</td>
<td>(Daily)</td>
</tr>
<tr>
<td>8am</td>
<td>VM bck to vault</td>
<td>(Daily)</td>
</tr>
</tbody>
</table>

## Dev and Test:

<table>
<thead>
<tr>
<th>Time</th>
<th>Task Description</th>
<th>Start/End Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>12am</td>
<td>Both Dev &amp; QA machines shut down</td>
<td>Manual start</td>
</tr>
<tr>
<td>1am</td>
<td>(when in use)</td>
<td></td>
</tr>
<tr>
<td>2am</td>
<td>Manual (on-demand) ETL &amp; SSAS processing</td>
<td></td>
</tr>
<tr>
<td>3am</td>
<td>EST standard business hours</td>
<td></td>
</tr>
<tr>
<td>4am</td>
<td>UK standard business hours</td>
<td></td>
</tr>
<tr>
<td>5am</td>
<td>Auto patching</td>
<td>(Fri)</td>
</tr>
<tr>
<td>6am</td>
<td>SSIS maint</td>
<td>(Daily)</td>
</tr>
<tr>
<td>7am</td>
<td>SQL backup</td>
<td>(Daily)</td>
</tr>
<tr>
<td>8am</td>
<td>VM bck to vault</td>
<td>(Daily)</td>
</tr>
<tr>
<td>9am</td>
<td>Auto patching: requires VM to be on.</td>
<td></td>
</tr>
<tr>
<td>10am</td>
<td>VM backup to vault: not require VM to be on</td>
<td></td>
</tr>
</tbody>
</table>
Automatic Shutdown

Dev & Test
We are using the auto-shutdown feature within the VM properties (Dev/Test Labs).

Production
We are using Azure Automation to shut down, and restart, the 1st VM (DB engine/SSIS) overnight since our ETL runs 1x/hour during business hours only. This saves ~33% of compute cost.

The 2nd VM (SSAS) stays on 24/7.
Automatic Patching

We are using the Windows and SQL auto-patching feature. The VM “Manage Updates” feature is in private preview (July 2017).

VM Backups

The VM backups are managed in the Recovery Services Vault.

For the VM backups to be encrypted, disk encryption is required (discussed in the previous section).
SQL Backups: From VM

We are using the scheduled SQL Server backups feature.

Be sure that this storage account is set to be encrypted – the “automaticbackup” container contains certificates & keys.
SQL Backups: From SQLDB

Backups for SQLDB depend on the pricing tier selected:

Full backups: weekly
Differential: every few hours
T-log: every 5-10 minutes

Basic service: retained 7 days
Standard service: retained 35 days
Premium service: retained 35 days

Longer-term backup retention:
• Set up a process to back up the .bacpac file (schema + data), or
• Recovery Services vault (still in Preview):

More info: https://blogs.msdn.microsoft.com/mast/2013/03/03/different-ways-to-backup-your-windows-azure-sql-database/
ARM Templates

Goal is for deployments to be modular & repeatable.

We have invested time in learning how to do ARM templates so we can make a good decision when to use them.

We have our “infrastructure as code” checked into source control within a Visual Studio project.

We are being selective about automation.
Balance of: wanting infrastructure in source code vs. little need for deployment repeatability.

We have some PowerShell scripts for deployment purposes, such as:
- Assign tags
- Enable disk encryption
- Provision key vault
- Deploy ARM template

Blog post on ARM deployments:
Recommendations for ARM and PowerShell

Use ARM for:
• Deployment of resources to Test and Prod
  • Recognition of dependencies
  • Parallel deployment of resources (faster than PowerShell or CLI)

Use PowerShell for:
• Execution of ARM template
• Management of resources (ex: start/stop a VM - in conjunction with Azure Automation)
• Selective items you would prefer in PowerShell vs. ARM (ex: Tags or Metrics Alerts)
Monitoring the Solution
Monitoring

We are currently using a combination of:

- Azure Monitor (Metrics, Activity Log, Diagnostic Log)
- Azure Monitor Alerts
- Azure Blob Storage (Metrics & Activity Log Data)
- Azure Security Center
- SentryOne software

We have not needed Microsoft OMS (Operations Management Suite) as of yet, but we may grow into it.
Azure Monitor

1. Metrics
   ✓ Performance counters in 1-minute frequency

2. Activity Log
   ✓ Captures create, update & delete for resources
   ✓ Still called ‘Operational Logs’ when sent to blob storage

3. Diagnostic Log
   ✓ Emitted by each resource
Azure Monitor: Alerts

**Metric Alert**
- Email or webhook specified within the alert
- Always applies to one single resource

**Activity Log Alert**
- Can use an Action Group for SMS, email, or webhook
- Is a resource itself assigned to a RG
- Can span resource groups & resources

There are currently no alerts for the diagnostics logs.
Azure Monitor: Action Groups

Used with Activity Log Alerts only right now (not supported with Metric Alerts).

Action types:
- SMS
- Email
- Webhook

Generally only Prod resources have any alerting configured.

Critical alerts: SMS text + e-mail
Non-critical alerts: e-mail only
Azure Diagnostics: SQL Database (PaaS)

We are sending diagnostics to:
• Storage Account (JSON data)
• Log Analytics

Partitioned in Storage as low as:
Resource Type > Year > Month > Day
> Hour > Minute
Auditing is set at both the Server and the Database level for SQLDB.

Send to: the ‘diagnostics’ blob storage account associated with this particular project.
Performance Counters
SQL Server counters are not gathered by default – need to explicitly select them.

Agent
The Azure Diagnostics Agent doesn’t prompt for storage account – will need to change (i.e., if you have a naming convention for diagnostics storage), and then delete the auto-created storage account.
Azure Security Center

SentryOne Monitoring

Support for monitoring:
✓ SQL Server
  (on-prem, or in Azure or a cloud VM)
✓ Azure SQL Database
✓ Azure SQL Data Warehouse
✓ SQL Server Analysis Services
  (on-prem, or in Azure or a cloud VM)
✓ Windows Computer
✓ Tintri VMstore
✓ VMware Host
Key Takeaways and Q&A
Key Takeaways

The pace of change in Azure is frequent—plan to keep up with announcements. New features can change a previous decision really fast.

Look at using PaaS unless you really need an IaaS solution. Deploy the simplest solution that gets the job done.

Don’t shortcut the initial planning phase – do a POC if you can to learn. Don’t assume a feature is definitely available or works as you expect it to.

For unfamiliar features or services, consider creating a sandbox area to test & learn on first.

Plan to invest some time learning ARM templates & PowerShell, if you haven’t already.

Pay attention to efficiency and opportunities for cost savings. Developers & Administrators have a huge impact on cost.
Thank You!

To download a copy of this presentation:
SQLChick.com “Presentations & Downloads” page

Melissa Coates
Lead Analytics Architect,
SentryOne

Blog: sqlchick.com
Twitter: @sqlchick

Creative Commons License:
Attribution-NonCommercial-NoDerivative Works 3.0